A Recursive Markov Boundary-Based Approach to Causal Structure Learning

Ehsan Mokhtarian[*]   EHSAN.MOKHTARIAN@EPFL.CH
Department of Computer and Communication Science
EPFL, Lausanne, Switzerland

Sina Akbari[*]   SINA.AKBARI@EPFL.CH
Department of Computer and Communication Science
EPFL, Lausanne, Switzerland

AmirEmad Ghassami   AGHASSA1@JHU.EDU
Department of Computer Science
Johns Hopkins University, Baltimore, USA

Negar Kiyavash   NEGAR.KIYAVASH@EPFL.CH
College of Management of Technology
EPFL, Lausanne, Switzerland

Abstract

Constraint-based methods are one of the main approaches for causal structure learning that are particularly valued as they are asymptotically guaranteed to find a structure that is Markov equivalent to the causal graph of the system. On the other hand, they may require an exponentially large number of conditional independence (CI) tests in the number of variables of the system. In this paper, we propose a novel recursive constraint-based method for causal structure learning that significantly reduces the required number of CI tests compared to the existing literature. The proposed approach aims to use Markov boundary information to identify a specific variable that can be removed from the set of variables without affecting the statistical dependencies among the other variables. Having identified such a variable, we discover its neighborhood, remove that variable from the set of variables, and recursively learn the causal structure over the remaining variables. We further provide a lower bound on the number of CI tests required by any constraint-based method. Comparing this lower bound to our achievable bound demonstrates the efficiency of the proposed approach. Our experimental results show that the proposed algorithm outperforms state-of-the-art both on synthetic and real-world structures.

Keywords: Causal Discovery, Recursive Causal Structure Learning, Bayesian Networks

1. Introduction

Learning the causal structure among the variables of the system under study is one of the main goals in many fields of science. This task has also gained significant attention in the recent three decades in artificial intelligence because it has become evident that the

---

[*]. Equal contribution.

knowledge of causal structure can significantly improve the prediction power and remove systematic biases in inference (Pearl, 2009; Spirtes et al., 2000).

One of the main assumptions in causal structure learning is that the ground truth structure is a directed acyclic graph (DAG). There are two main classes of methods in the literature for learning the causal DAG, namely constraint-based methods and score-based methods (Zhang et al., 2017). The idea in constraint-based methods is to find the most consistent structure with the conditional independence relations in the data. The most well-known constraint-based method is the PC algorithm (Spirtes et al., 2000). In score-based causal structure learning, the idea is to search for a structure that maximizes a score function, commonly chosen to be a regularized likelihood function. The search for the optimum structure is usually based on a greedy search (Heckerman et al., 1995; Chickering, 2002; Teyssier and Koller, 2012; Solus et al., 2017), or continuous optimization Zheng et al. (2018); Manzour et al. (2021). There are also hybrid methods that combine constraint-based and score-based methods (Tsamardinos et al., 2006), as well as methods that require specific assumptions on the data generating modules, such as requiring linearity and non-Gaussianity of the noises (Shimizu et al., 2006) or additivity of the noise with specific types of non-linearity (Hoyer et al., 2009).

Constraint-based methods are particularly valued as they do not require any assumptions on the functional form of the causal modules and recover a structure that is Markov equivalent to the causal graph of the system. One of the caveats to these methods is that in the worst case, the number of conditional independence (CI) tests required can be exponentially large in the number of variables. Several efforts in the literature have tried to reduce the number of required CI tests and improve the performance of constraint-based methods (Margaritis and Thrun, 1999; Kalisch and Bühlmann, 2007; Pellet and Elisseeff, 2008; Xie and Geng, 2008; Zhang et al., 2019).

(Spirtes et al., 2000) proposed the so-called PC algorithm with complexity $\mathcal{O}(p^\Delta)$, where p and Δ denote the number of variables and the maximum degree of the underlying graph, respectively. (Margaritis and Thrun, 1999) and (Pellet and Elisseeff, 2008) proposed using Markov boundary information to reduce the number of required CI tests. The former proposed the GS method with complexity $\mathcal{O}(p^2 + p\alpha^2 2^\alpha)$, and the latter proposed the CS method with complexity $\mathcal{O}(p^2 2^\alpha)$, where α is the maximum size of the Markov boundary among the variables. To the best of our knowledge, these are the state-of-the-art achievable bounds in the literature.

In this paper, we propose a novel recursive constraint-based method for causal structure learning, which we call *MARVEL*. Our method is non-parametric and does not posit any assumptions on the functional relationships among the variables, while it significantly reduces the number of required CI tests. In each iteration of our recursive approach, we use the Markov boundary information to find a *removable* variable (see Definition 3). We then orient the edges incident to this variable and remove it from the set of variables. Finally, we update the Markov boundary information for the next iteration.

Our main contributions are as follows.

- We introduce the notion of a *removable* variable, which is a variable that can be removed from a DAG without changing the d-separation relations (Definition 3). Moreover, we provide a graphical characterization of removable variables (Theorem 5).
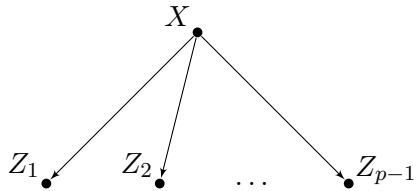
Figure 1: A DAG with $\Delta_{\text{in}} = 1$ and $\Delta = \alpha = p - 1$.

- Harnessing the notion of removability, we propose a novel recursive Markov boundary-based causal structure learning method, called MARVEL (Section 3). Given the Markov boundary information, MARVEL requires $\mathcal{O}(p\Delta_{\text{in}}^2 2^{\Delta_{\text{in}}})$ CI tests in the worst case to find the Markov equivalence class of the causal graph, where p and Δ_{in} are the number of variables and the maximum in-degree of the causal DAG, respectively (Proposition 14). We show that this upper bound significantly improves over the state of the art.

- We provide a lower bound on the required number of CI tests for any constraint-based method. Specifically, we show that any constraint-based algorithm requires at least $\Omega(p^2 + p\Delta_{\text{in}} 2^{\Delta_{\text{in}}})$ CI tests in the worst case (Theorem 16). Comparing this lower bound with our achievable bound demonstrates the efficiency of our proposed method.

In Section 2.1 we discuss well-known Markov boundary discovery algorithms which require $\mathcal{O}(p^2)$ CI tests. By utilizing one of these algorithms, our method discovers the causal graph by performing $\mathcal{O}(p^2 + p\Delta_{\text{in}}^2 2^{\Delta_{\text{in}}})$ CI tests in the worst case. It is noteworthy that our upper bound is based on Δ_{in} as opposed to Δ or α. We achieve this by the virtue of recursive variable elimination. In general $\Delta_{\text{in}} \leq \Delta \leq \alpha$. Additionally, in a DAG with a constant Δ_{in}, the values of Δ and α can grow linearly with the number of variables. Figure 1 depicts one such graph where $\Delta_{\text{in}} = 1$, but $\Delta = \alpha = p - 1$. Therefore, in some cases PC, GS, and CS requires an exponential number of CI tests while our approach has merely quadratic complexity. Our experiments on both synthetic and real-world structures show that MARVEL requires substantially fewer CI tests with smaller average size of conditioning sets while obtaining superior accuracy, compared to state-of-the-art constraint-based methods.

We start the exposition by introducing the notations, reviewing the terminology and describing the problem in Section 2. MARVEL method is described in Section 3, and its computational complexity is discussed in Section 4. Section 5 is dedicated to evaluating MARVEL on both synthetic and real-world structures.

2. Preliminaries and Problem Description

We consider a system with p variables denoted by the set $\mathbf{V}$. Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be the directed acyclic graph (DAG) over $\mathbf{V}$ which represents the causal relationships among the variables, where $\mathbf{E}$ is the set of directed edges. A directed edge from variable X to Y, denoted by (X, Y), represents that X is a direct cause of Y with respect to $\mathbf{V}^1$. This model is referred to as causal DAG or causal Bayesian network in the literature (Pearl, 2009; Spirtes et al.,

---

1. Through out the paper, we use the terms variable and vertex interchangeably.

2000; Neapolitan, 2004). The *skeleton* of $\mathcal{G}$ is defined as the undirected graph obtained by removing the directions of the edges of $\mathcal{G}$. If $(X, Y) \in E$, X and Y are called *neighbors* of each other, X is a *parent* of Y and Y is a *child* of X. The set of all neighbors, parents, and children of X are denoted by N_X, Pa_X, and Ch_X, respectively.

Definition 1 (v-structure) *Three vertices form a* v-structure *if two of them are parents of the third vertex while they are not neighbors themselves. Additionally, $\mathcal{V}_X^{Pa}$ denotes the set of v-structures in which X is a parent.*

Definition 2 (co-parent) *For $X, Y \in \mathbf{V}$, Y is a co-parent of X if it shares at least one child with X and $Y \notin N_X$. The set of co-parents of X is denoted by Λ_X.*

A distribution $P_{\mathbf{V}}$ on variables $\mathbf{V}$ satisfies *Markov property* with respect to $\mathcal{G}$ if d-separation[2] in $\mathcal{G}$ implies conditional independence (CI) in P_V. That is, X is d-separated from Y by $\mathbf{S} \subseteq \mathbf{V}$, denoted by $X \perp_{\mathcal{G}} Y|\mathbf{S}$, implies $X \perp\!\!\!\perp_{P_{\mathbf{V}}} Y|\mathbf{S}$. Conversely, $P_{\mathbf{V}}$ satisfies *faithfulness* with respect to $\mathcal{G}$, if CI in $P_{\mathbf{V}}$ implies d-separation in $\mathcal{G}$. That is, $X \perp\!\!\!\perp_{P_{\mathbf{V}}} Y|\mathbf{S}$ implies $X \perp_{\mathcal{G}} Y|\mathbf{S}$ (Spirtes et al., 2000; Glymour and Cooper, 1999). We often drop the subscripts $\mathcal{G}$ and $P_{\mathbf{V}}$ when there is no ambiguity.

In this paper we study the problem of causal structure learning under the Markov condition and faithfulness assumption. Additionally, we assume causal sufficiency, that is, we assume that the variables do not have any latent common causes. Under these assumptions, the underlying causal DAG can be learned up to Markov equivalence class, i.e., the set of DAGs representing the same conditional independence relationships (Spirtes et al., 2000; Pearl, 2009). (Verma and Pearl, 1991) showed that two DAGs are Markov equivalent if and only if they have the same skeleton and v-structures. The Markov equivalence class of a DAG can be graphically represented by a partially directed graph called the *essential graph*. Our goal is to obtain the essential graph corresponding to the causal DAG from observational data.

Before proceeding to our proposed approach, we briefly review a few definitions and results on Markov boundaries.

2.1 Markov Boundary

For $X \in \mathbf{V}$, Markov boundary of X is a minimal set $\mathbf{S} \subseteq \mathbf{V} \setminus \{X\}$ such that $X \perp\!\!\!\perp \mathbf{V} \setminus (\mathbf{S} \cup \{X\})|\mathbf{S}$. Under Markov and faithfulness assumptions, Markov boundary of each vertex X, denoted by Mb_X, is unique and consists of its parents, children and co-parents (Pearl, 1988, 2009):

$$Mb_X = \text{Pa}_X \cup \text{Ch}_X \cup \Lambda_X = N_X \cup \Lambda_X. \tag{1}$$

Many algorithms have been proposed in the literature for discovering the Markov boundaries (Fu and Desmarais, 2010; Margaritis and Thrun, 1999; Guyon et al., 2002; Tsamardinos and Aliferis, 2003; Tsamardinos et al., 2003b; Yaramakala and Margaritis, 2005). One straightforward method is *total conditioning* (TC) (Pellet and Elisseeff, 2008), which states that under faithfulness, X and Y are in each other's Markov boundary if and only if

$$X \not\perp\!\!\!\perp Y|\mathbf{V} \setminus \{X, Y\}. \tag{2}$$

---

2. See (Pearl, 1988) for the definition of d-separation.

Using total conditioning, $\binom{|\mathbf{V}|}{2}$ CI tests suffice for identifying the Markov boundaries of all of the vertices. However, the drawback is that each CI test requires conditioning on a large set of variables.

This issue is addressed in several algorithms including Grow-Shrink (GS) (Margaritis and Thrun, 1999), IAMB (Tsamardinos et al., 2003b) and its several variants which propose a method that leads to performing more CI tests, but with smaller conditioning sets. Thus, the choice of which algorithm to use for computing the Markov boundaries must be made according to the data at hand. Note that these algorithms perform at most $\mathcal{O}(p^2)$ CI tests to discover the Markov boundaries. The Markov boundary information is required to initialize our proposed approach, and any of these methods can be utilized for this purpose.

3. MARVEL Method

In this section, we present our recursive method for learning the causal structure. The idea in this approach is as follows.

We first identify a variable with certain properties, which we call *removable* using Markov boundary information. A removable variable is a variable that can be omitted from the causal graph such that the remaining graph still satisfies Markov property and faithfulness with respect to the marginal distribution of the remaining variables. We then identify the neighbors of this variable and orient the edges incident to it. Finally, we remove this variable from the set of variables and update the Markov boundaries for the next iteration. This procedure is repeated until all of the variables are removed.

We show that this approach leads to finding a graph with the same skeleton and v-structures as the true causal graph. Hence, the corresponding essential graph can be identified using this graph. We call our method, *Markov boundary-based Recursive Variable Elimination (MARVEL)*.

We introduce removable variables in Subsection 3.1, propose a method for testing removability in Subsection 3.2, and present our recursive algorithm in Subsection 3.3. In Subsection 3.4 we show how to avoid performing duplicate CI tests during our algorithm.

3.1 Removable Variables

A removable variable is defined formally as follows.

Definition 3 (Removable) X *is a removable vertex in a DAG $\mathcal{G}$ if the d-separation relations in $\mathcal{G}$ and $\mathcal{H} := \mathcal{G} \setminus \{X\}$ are equivalent over the vertices of $\mathcal{H}$. That is for any vertices $Y, Z \in \mathbf{V} \setminus \{X\}$ and $\mathbf{S} \subseteq \mathbf{V} \setminus \{X, Y, Z\}$,*

$$Y \perp_{\mathcal{G}} Z | \mathbf{S} \iff Y \perp_{\mathcal{H}} Z | \mathbf{S}. \tag{3}$$

Remark 4 *Suppose $P_{\mathbf{V}}$ is Markov and faithful with respect to a DAG $\mathcal{G}$. For any vertex $X \in \mathbf{V}$, $P_{\mathbf{V} \setminus \{X\}}$ is Markov and faithful with respect to $\mathcal{G} \setminus \{X\}$ if and only if X is a removable vertex in $\mathcal{G}$.*

All the proofs are available in Appendix A. Next, we propose a graphical characterization of removable variables.

Theorem 5 (Removability) X *is removable in* $\mathcal{G}$ *if and only if the following two conditions are satisfied for every* $Z \in Ch_X$.

Condition 1: $N_X \subset N_Z \cup \{Z\}$.

Condition 2: $Pa_Y \subset Pa_Z$ *for any* $Y \in Ch_X \cap Pa_Z$.

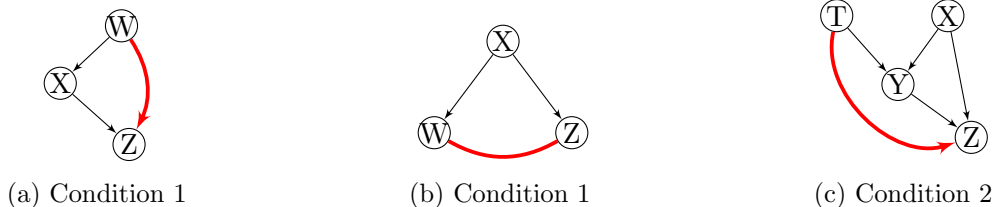(a) Condition 1   (b) Condition 1   (c) Condition 2

Figure 2: Conditions of Theorem 5.

Figure 2 depicts the two Conditions of Theorem 5. In Figures 2a and 2b, $W \in N_X$. Hence, Condition 1 implies the red edge between W and Z. Note that W must be a parent of Z in Figure 2a since $\mathcal{G}$ is a DAG. In Figure 2c, $T \in Pa_Y$. Hence, Condition 2 implies the red edge between T and Z.

Remark 6 *Variables with no children satisfy the two conditions of Theorem 5. Therefore, there always exists a removable vertex in every DAG.*

3.2 Testing for removability

In this section, we present an approach for testing the removability of a variable given the Markov boundary information of all of the variables when Markov and faithful assumptions hold, i.e., when d-separation is equivalent to CI. Our method is based on evaluating the two conditions of Theorem 5. In Subsections 3.2.1 and 3.2.2 we discuss how to identify the neighbors, co-parents, and $\mathcal{V}_X^{\mathrm{pa}}$ efficiently. Subsequently, we use this information to test Conditions 1 and 2 of Theorem 5 in Subsections 3.2.3 and 3.2.4.

3.2.1 Finding Neighbors and co-parents of a variable

Given the Markov boundary of a variable, we can use the following lemma to tell the neighbors and co-parents apart (Margaritis and Thrun, 1999).

Lemma 7 *Suppose* $X \in \mathbf{V}$ *and* $Y \in Mb_X$. Y *is a neighbor of* X *if and only if*

$$X \not\perp\!\!\!\perp Y | \mathbf{S}, \quad \forall \mathbf{S} \subset Mb_X \setminus \{Y\}. \tag{4}$$

Given Mb_X, for each $Y \in Mb_X$ we perform the CI tests of Equation 4. If all of theses tests yield dependence, then X and Y are neighbors. Otherwise, there exists $\mathbf{S}_{XY} \subset Mb_X \setminus \{Y\}$ that d-separates X and Y. Hence, we can identify N_X and Λ_X (along with a separating set for each co-parent) with at most $|Mb_X|2^{|Mb_X|-1}$ CI tests.

3.2.2 Finding v-structures ($\mathcal{V}_X^{\text{pa}}$)

Recall that $\mathcal{V}_X^{\text{pa}}$ denotes the set of v-structures in which X is a parent.

Lemma 8 *Suppose $T \in \Lambda_X$ with a separating set $\mathbf{S}_{XT}$ for X and T, and let $Y \in N_X$. Y is a common child of X and T (i.e., $X \to Y \leftarrow T$ is in $\mathcal{V}_X^{pa}$) if and only if $Y \notin \mathbf{S}_{XT}$ and*

$$Y \not\perp\!\!\!\perp T | \mathbf{S}, \quad \forall \mathbf{S} \subseteq Mb_X \cup \{X\} \setminus \{Y, T\}. \tag{5}$$

Once N_X and Λ_X are identified along with a separating set for each of the variables in Λ_X using Lemma 7, we can find $\mathcal{V}_X^{\text{pa}}$ by applying Lemma 8 which requires performing at most $|\Lambda_X||N_X|2^{|Mb_X|-1}$ CI tests.

3.2.3 Testing condition 1

To test condition 1 of Theorem 5 for a variable X, we apply the following lemma.

Lemma 9 *Variable X satisfies Condition 1 of Theorem 5 if and only if*

$$Z \not\perp\!\!\!\perp W | \mathbf{S} \cup \{X\}, \quad \forall W, Z \in N_X, \mathbf{S} \subseteq Mb_X \setminus \{Z, W\}. \tag{6}$$

Given N_X, Condition 1 for X can be verified using Lemma 9 by performing at most $\binom{|N_X|}{2}2^{|Mb_X|-2}$ CI tests.

3.2.4 Testing condition 2

Lemma 10 *Suppose the variable X satisfies Condition 1 of Theorem 5. Then X satisfies Condition 2 of Theorem 5, and therefore, X is removable, if and only if*

$$Z \not\perp\!\!\!\perp T | \mathbf{S} \cup \{X, Y\}, \quad \forall (X \to Y \leftarrow T) \in \mathcal{V}_X^{pa}, Z \in N_X \setminus \{Y\}, \mathbf{S} \subseteq Mb_X \setminus \{Z, Y, T\}. \tag{7}$$

Having identified N_X and $\mathcal{V}_X^{\text{pa}}$, Lemma 10 allows us to verify if Condition 2 holds for X using at most $|\Lambda_X||N_X|2^{|Mb_X|-2}$ unique CI tests, since $Z \in N_X$, $T \in \Lambda_X$, and $\mathbf{S} \cup \{Y\} \subseteq Mb_X \setminus \{Z, T\}$. In Section 3.4 we show how to perform these CI tests without performing duplicate tests. The following proposition summarizes the results of this section.

Proposition 11 *Knowledge of Mb_X suffices to identify $N_X, \Lambda_X, \mathcal{V}_X^{pa}$, and determine whether X is removable by performing at most $\mathcal{O}(|Mb_X|^2 2^{|Mb_X|})$ unique CI tests.*

3.3 Algorithm

In order to identify a removable vertex efficiently (i.e. in terms of the number of CI tests), we sort the variables based on the cardinality of their Markov boundary in ascending order. Let $\mathcal{I} = (X_{(1)}, X_{(2)}, ..., X_{(p)})$ be this ordering. Starting with $X_{(1)}$, we look for the first variable $X_{(i)}$ that is removable. Testing the removability of each variable is performed through the CI tests described in the previous section, and we stop when we identify the first removable variable in this order. The following lemma guarantees that the Markov boundary of every variable for which we perform the removability tests until we reach the first one to prove removable is at most of size Δ_{in}.

Algorithm 1: MARVEL

1: **Input: V**, $P_{\mathbf{V}}$, $(Mb_X \colon X \in \mathbf{V})$

2: Create $\hat{\mathcal{G}} = (\mathbf{V}, \mathbf{E} = \varnothing)$ with vertex set $\mathbf{V}$ and no edges.

3: $\overline{\mathbf{V}} \leftarrow \mathbf{V}$ % $\overline{V}$ is the set of remaining variables.

4: **for** 1 **to** $|\mathbf{V}|$ **do**

5:      $\mathcal{I} = (X_{(1)}, X_{(2)}, ..., X_{(|\overline{\mathbf{V}}|)}) \leftarrow$ Sort $\overline{\mathbf{V}}$ in ascending order based on their Markov boundary size.

6:      **for** $i = 1$ **to** $|\overline{\mathbf{V}}|$ **do**

7:          $(N_{X_{(i)}}, \Lambda_{X_{(i)}}, \mathcal{S}_{X_{(i)}}) \leftarrow$ Find neighbors and co-parents of $X_{(i)}$ along with a set of separating sets $\mathcal{S}_{X_{(i)}} = (\mathbf{S}_{X_{(i)}Y} \colon Y \in \Lambda_{X_{(i)}})$ using Lemma 7.

8:          Add undirected edges between $X_{(i)}$ and $N_{X_{(i)}}$ to $\hat{\mathcal{G}}$ if an edge is not already there.

9:          **if** Equation 6 holds for $X = X_{(i)}$ **then**

10:             Find $\mathcal{V}_{X_{(i)}}^{\mathrm{pa}}$ using Lemma 8.

11:             Orient the edges of $\mathcal{V}_{X_{(i)}}^{\mathrm{pa}}$ accordingly in $\hat{\mathcal{G}}$.

12:             **if** Equation 7 holds for $X = X_{(i)}$ **then**

13:                 Orient the remaining undirected edges incident to $X_{(i)}$ in $\hat{\mathcal{G}}$ as in-going towards $X_{(i)}$.

14:                 $\overline{\mathbf{V}} \leftarrow \overline{\mathbf{V}} \setminus \{X_{(i)}\}$

15:                 Update Mb_Y for all $Y \in Mb_{X_{(i)}}$.   % See Subsection 3.3.1.

16:                 **Break** the for loop of line 6.

17: $\tilde{\mathcal{G}} \leftarrow$ The partially directed graph with the skeleton and the v-structures of $\hat{\mathcal{G}}$.

18: Apply the Meek rules (Meek, 1995) to $\tilde{\mathcal{G}}$.

19: **return** $\tilde{\mathcal{G}}$

Lemma 12 *If $X \in \mathbf{V}$ is a removable vertex in $\mathcal{G}$, then $|Mb_X| \leq \Delta_{in}$, where Δ_{in} is the maximum in-degree of $\mathcal{G}$.*

As shown in Section 3.2, the CI tests required for both learning the neighbors of a variable and testing its removability are conditioned on a subset of its Markov boundary. Since vertices are processed in the order $\mathcal{I}$ up to a removable variable, by Lemma 12 we only process the variables with a Markov boundary of size at most Δ_{in}. This ensures a maximum size of conditioning sets, which in turn results in more powerful conditional independence tests, and also results in performing substantially fewer CI tests to learn the structure. Note that if we recursively eliminate the first removable variable in $\mathcal{I}$ at each iteration, the succeeding variables in $\mathcal{I}$, which are currently not removable, will eventually become removable at some point. Therefore, during the process, we never need to learn the neighborhood of a variable with Markov boundary size larger than Δ_{in}. We discuss our complexity upper bound in Section 4. The pseudo code of MARVEL is outlined in Algorithm 1.

Initially, the output of a Markov boundary discovery algorithm (e.g., one of the algorithms mentioned in Section 2.1) is input to the MARVEL method. The procedure is then initialized with an empty graph $\hat{\mathcal{G}}$ (line 2). $\overline{\mathbf{V}}$ indicates the set of remaining variables. Each iteration of the recursive part consists of three main phases:

1. Identify the first removable vertex $X_{(i)}$ in $\mathcal{I}$.

2. Discover $N_{X_{(i)}}$ and orient the edges incident to $X_{(i)}$.

3. Remove $X_{(i)}$ and update the Markov boundary of the remaining variables.

In phase 1, we first obtain $\mathcal{I}$ in line 5. We then check the two condition of removability using Lemmas 9 and 10 in lines 9 and 12. Phase 2 is performed in the following steps. First, we learn $N_{X_{(i)}}$ and add the corresponding undirected edges to $\hat{\mathcal{G}}$ if the edge is not already added (lines 7, 8). We then orient the v-structure edges of $\mathcal{V}_X^{\mathrm{pa}}$ (line 10). Finally, we orient the remaining undirected edges incident to $X_{(i)}$ as in-going towards $X_{(i)}$ (line 13). We will show in Theorem 13 that with this orientation, $\hat{\mathcal{G}}$ will have the same v-structures as the causal graph $\mathcal{G}$. In phase 3, $X_{(i)}$ is removed from the graph, and we update the Markov boundaries of the remaining variables in the absence of $X_{(i)}$ (lines 14, 15). In Subsection 3.3.1 we will present an efficient method for the latter task. Note that $X_{(i)}$ is particularly chosen such that the marginal distribution of the remaining variables satisfies faithfulness with respect to the induced subgraph of $\mathcal{G}$ over these variables.

At the end of the algorithm, we keep the direction of the edges involved in v-structures in $\hat{\mathcal{G}}$ and apply the Meek rules (Meek, 1995) to this partially directed graph. Note that initially, we assumed Markov property and faithfulness. Although we remove a variable at each iteration, Remark 4 implies that Markov and faithfulness assumptions hold in all iterations since we remove removable variables. Hence, throughout the execution of the algorithm, the CI tests are equivalent to the d-separation relations in the remaining graph.

The theorem below provides the correctness of MARVEL method.

Theorem 13 (Correctness of MARVEL) *Suppose $\mathcal{G}$ satisfies Markov property and faithfulness with respect to $P_{\mathbf{V}}$. The learned graph $\hat{\mathcal{G}}$ in Algorithm 1 has the same skeleton and v-structures as $\mathcal{G}$. Therefore, the output of Algorithm 1 is the essential graph corresponding to $\mathcal{G}$.*

3.3.1 Updating Markov boundaries

Suppose X is removed in an iteration. At the end of this iteration, we need to update the Markov boundaries of the remaining vertices. The removal of X affects the remaining vertices in two ways:

1. X should be removed from Markov boundaries of every vertex it appeared in.

2. If two variables are not adjacent and have only X as a common child, then they should be removed from each other's Markov boundary.

Consequently, it is sufficient to remove X from the Markov boundary of the vertices in Mb_X, and then update the Markov boundary only for the vertices in N_X, i.e., the only vertices that can potentially be parents of X. Therefore, it suffices that for all pairs $\{Y, Z\}$ in N_X we check whether Y and Z remain in each other's Markov boundary after removing X. This is equivalent to testing for the dependency $Y \not\perp\!\!\!\perp Z | Mb_Z \setminus \{X, Y, Z\}$, or alternatively $Y \not\perp\!\!\!\perp Z | Mb_Y \setminus \{X, Y, Z\}$. We perform the test using the smaller of these two conditioning sets. Formally, define $W := \arg\min_{U \in \{Y,Z\}} |Mb_U|$. The test will check whether

$$Y \perp\!\!\!\perp Z | Mb_W \setminus \{X, Y, Z\}. \tag{8}$$

If Equation 8 holds, we remove Y from Mb_Z and Z from Mb_Y.

3.4 Avoiding duplicate CI tests

In an iteration of MARVEL, if a variable does not pass the removability tests, it will be tested for removability in the subsequent iterations. We discuss an approach to use the information from previous iterations to avoid performing duplicate CI tests while testing the removability of such variables.

3.4.1 CI tests for finding neighbors, co-parents, and v-structures

We identify neighbors and co-parents of a variable in line 7 of Algorithm 1 as described in 3.2.1. It suffices to do this procedure once for each variable X. More precisely, if we learn N_X in an iteration, the neighbors of X in the next iterations would be N_X excluding the deleted variables. Hence, we do not need any further CI tests for finding neighbors and co-parents of X in the following iterations.

The same applies to v-structures. If we find $\mathcal{V}_X^{\mathrm{pa}}$ in an iteration, we can save it and delete a v-structure from it when one of the three variables of the v-structure is removed.

3.4.2 CI tests for condition 1

For an arbitrary variable X, suppose $Z, W \in N_X$. If in an iteration, there does not exist any $\mathbf{S} \subseteq Mb_X \setminus \{Z, W\}$ such that $Z \perp\!\!\!\perp W | \mathbf{S} \cup \{X\}$, then either Z, W are neighbors or they are both parents of X. Since removing a vertex does not alter such relationships, no $\mathbf{S} \cup \{X\}$ can separate W, Z in the following iterations either. We can use this information to skip performing duplicate CI tests for pairs $W, Z \in N_X$ in the following iterations.

3.4.3 CI tests for condition 2

The same idea applies to condition 2. Suppose $Z \in N_X$ and $X \to Y \leftarrow T$ is in $\mathcal{V}_X^{\mathrm{pa}}$. If in an iteration for all $\mathbf{S} \subseteq Mb_X \setminus \{Z, Y, T\}$, $Z \not\perp\!\!\!\perp T | \mathbf{S} \cup \{X, Y\}$, then Z, T do not have any separating set including X, Y. Hence, we can save this information and avoid performing CI tests for the pair Z, T when the separating set includes both X, Y.

4. Complexity Analysis

The bottleneck in the complexity of constraint-based causal structure learning methods is the number of CI tests they perform. In this section, we provide an upper bound on the worst-case complexity of the MARVEL algorithm in terms of the number of CI tests and compare it with PC (Spirtes et al., 2000), GS (Margaritis and Thrun, 1999), CS (Pellet and Elisseeff, 2008), and MMPC (Tsamardinos et al., 2003a) algorithms. We also provide a worst-case lower bound for any constraint-based algorithm to demonstrate the efficiency of our approach.

Proposition 14 *Given the initial Markov boundaries, the number of CI tests required by Algorithm 1 on a graph of order p and maximum in-degree Δ_{in} is upper bounded by*

$$p\binom{\Delta_{in}}{2} + \frac{p}{2}\Delta_{in}(1 + 0.45\Delta_{in})2^{\Delta_{in}} = \mathcal{O}(p\Delta_{in}^2 2^{\Delta_{in}}). \tag{9}$$

Corollary 15 *If $\Delta_{in} \leq c \log p$, MARVEL uses at most $\mathcal{O}(p^{c+1} \log^2 p)$ CI tests in the worst case, which is polynomial in the number of variables.*

If we use any of the algorithms mentioned in Section 2.1 that compute the Markov boundaries with $\mathcal{O}(p^2)$ CI tests, the overall upper bound of MARVEL on the number of CI tests will be

$$\mathcal{O}(p^2 + p\Delta_{in}^2 2^{\Delta_{in}}). \tag{10}$$

Theorem 16 *The number of conditional independence tests of the form $X \perp\!\!\!\perp Y|\mathbf{S}$ required by any constraint-based algorithm on a graph of order p and maximum in-degree Δ_{in} in the worst case is lower bounded by*

$$\Omega(p^2 + p\Delta_{in} 2^{\Delta_{in}}). \tag{11}$$

For constant Δ_{in} or more generally $\Delta_{in} = o(\log p)$, Equations 10 and 11 are quadratic in p. For larger values of Δ_{in}, the upper bound differs from the lower bound by a factor of Δ_{in}.

Table 1: Number of required CI tests in the worst case by various algorithms for causal structure learning.

Algorithm	Number of CI tests in the worst case
PC[3]	$\mathcal{O}(p^\Delta)$
GS	$\mathcal{O}(p^2 + p\alpha^2 2^\alpha)$
CS	$\mathcal{O}(p^2 2^\alpha)$
MMPC	$\mathcal{O}(p^2 2^\alpha)$
MARVEL	$\mathcal{O}(p^2 + p\Delta_{in}^2 2^{\Delta_{in}})$
Lower bound	$\Omega(p^2 + p\Delta_{in} 2^{\Delta_{in}})$

Table 1 compares the complexity of various algorithms, where Δ and α denote the maximum degree and the maximum Markov boundary size of the causal graph, respectively. In general $\Delta_{in} \leq \Delta \leq \alpha$. Additionally, in a DAG with a constant in-degree, Δ and α can grow linearly with the number of variables. Therefore, not only MARVEL has a significantly smaller worst-case bound, but also the complexity bound for all other algorithms can be exponential in some regimes where MARVEL remains polynomial.

5. Experiments

We evaluate MARVEL and compare it with other methods in two settings[4]. In Subsection 5.1, we assess the complexity of various causal structure learning algorithms in terms of the number of CI tests and size of conditioning sets, given oracle CI tests, i.e., when algorithms have access to true conditional independence relations among the variables. This is similar to assuming that the size of the observed samples is large enough to recover the conditional independence relations without any error. In this case, all of the algorithms recover the

---

3. If PC priory knows the exact value of Δ_{in} as side information, its upper bound will be $\mathcal{O}(p^{\Delta_{in}})$.
4. All of the experiments were run in MATLAB on a MacBook Pro laptop equipped with a 1.7 GHz Quad-Core Intel Core i7 processor and a 16GB, 2133 MHz, LPDDR3 RAM.

essential graph corresponding to the causal graph. In Subsection 5.2, we evaluate the algorithms on finite sample data, where we compare both the complexity and the accuracy of the algorithms over a wide range of sample sizes.

Our comparison cohort includes the modified version of PC algorithm that starts from the moralized graph[5] instead of the complete graph (to make a fair comparison with algorithms that start with Markov boundary information) (Pellet and Elisseeff, 2008; Spirtes et al., 2000), GS (Margaritis and Thrun, 1999), CS (Pellet and Elisseeff, 2008), and MMPC (Tsamardinos et al., 2003a) algorithms. We use MATLAB implementations of PC and MMPC provided in (Murphy et al., 2001; Murphy and Rosen, 2019) and (Tsirlis et al., 2018), respectively. The implementation of MARVEL is available in https://github.com/Ehsan-Mokhtarian/MARVEL.

5.1 Oracle Setting

We use d-separation relations in the causal DAG as the oracle answers for the CI tests in this setting. This is equivalent to having access to the joint distribution of the variables instead of a finite number of samples. We report the number of performed CI tests and the size of conditioning sets which are the main factors determining the time complexity of the algorithms. Note that CI tests with smaller conditioning sets are more reliable.

We use two random graph models for generating the causal DAGs. The first model is the directed Erdős-Rènyi model $G(p, m)$ (Erdős and Rényi, 1960), which provides uniform distribution over DAGs with p vertices and m edges (The skeleton is first sampled from the undirected Erdős-Rènyi model, and then the edges are oriented with respect to a random ordering over the variables). As the second model, we use random DAGs with a fixed maximum in-degree Δ_{in}, which are generated as follows. We fix a random ordering of the variables. For each vertex in the graph, we choose Δ_{in} potential parents among the other variables uniformly at random. We then choose the parents that do not violate the ordering. This procedure yields a DAG.

Figure 3 shows the experimental results regarding the number of CI tests. Subfigures 3a and 3d depict the result for the fixed Δ_{in} model. Subfigures 3b and 3c depict the result for the Erdős-Rènyi model, where the density of a graph is defined as the number of edges divided by the maximum possible number of edges, $\binom{p}{2}$. Each point in the plots is obtained using 20 DAGs. The shaded bars denote the 80% confidence intervals (Campbell, 2020). We continued running the algorithms until their run time exceeded a certain threshold, as the parameters (size of the graph, density or Δ_{in}) grew. As seen in the figures, MARVEL requires substantially fewer CI tests with smaller conditioning sets, and outperforms all other algorithms, especially on the graphs with fixed max in-degree. Also, as witnessed by Subfigures 3a and 3d, even the worst-case upper bound of our proposed method is below the number of CI tests required by the other algorithms.

---

5. The moralized graph of a DAG is the undirected graph in which every vertex is connected to all the variables in its Markov boundary.

(a) Fixed number of vertices ($p = 25$)

(b) Fixed number of vertices ($p = 25$)

(c) Fixed density (density is 0.25)

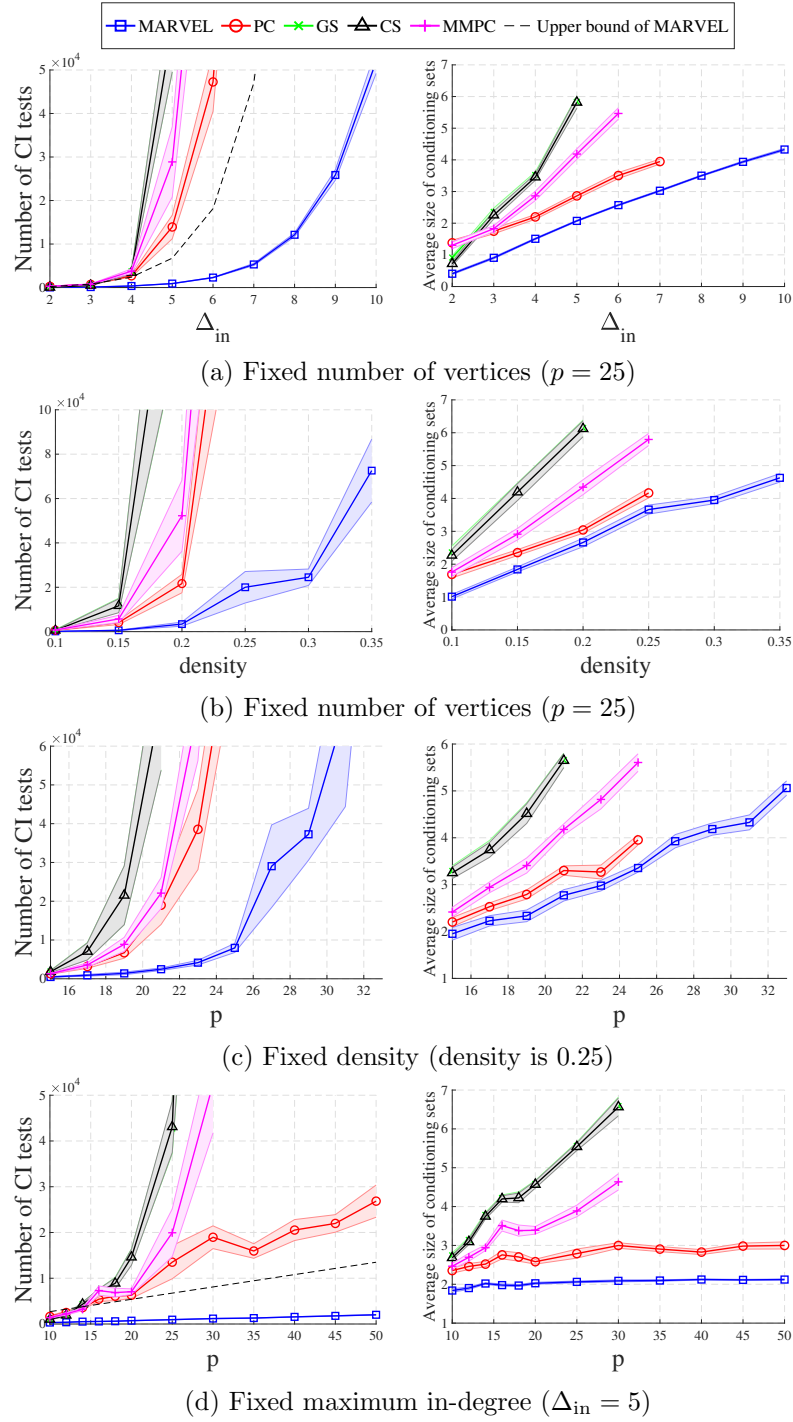(d) Fixed maximum in-degree ($\Delta_{\text{in}} = 5$)

Figure 3: Structure learning using oracle CI tests after Markov boundary discovery.

5.2 Finite Sample Setting

In this setting, we have access to a finite number of samples from $P_{\mathbf{V}}$. We evaluate the algorithms on various real-world structures available at Bayes Network Repository (Elidan, 2001) along with random graphs described in the last part.

We compare the algorithms in three scenarios.

5.2.1 Scenario 1: The effect of sample size on real-world structures

In this setting, we have picked 4 real-world structures, namely Insurance, Mildew, Barley, and Win95pts. After fixing the graph, the data is generated from a linear Gaussian structural causal model (Pearl, 2009), where each variable is generated as a linear combination of its parents plus a Gaussian noise. The coefficients are chosen uniformly at random from the interval $[-1, -0.5] \cup [0.5, 1]$ and the noise variables are distributed according to $\mathcal{N}(0, \sigma_X^2)$ where σ_X is chosen uniformly at random from the interval $[1, \sqrt{3}]$. The performance of the algorithms is measured by 1. the number of performed CI tests, 2. runtime[6], and 3. F1 score of the learned skeleton. We have used TC algorithm for Markov boundary discovery for all of the algorithms. Moreover, we have used Fisher Z-transformation (Fisher, 1915) to perform the CI tests with parameter $\alpha = \frac{2}{p^2}$, following the convention in (Pellet and Elisseeff, 2008), which includes an analysis over the choice of this parameter.

Figure 4 illustrates the results of this scenario. The reported results for the number of CI tests and runtime is after Markov boundary discovery. As seen in this figure, compared to the other algorithms, MARVEL is faster, requires a smaller number of CI tests, and obtains the highest accuracy in nearly all cases.

5.2.2 Scenario 2: random graphs

Data generating process in this section is the same as the previous scenario, and the algorithms are evaluated on a set of larger graphs.

Table 2 compares various algorithms on medium to large sized random graphs with $\Delta_{\text{in}} = 4$, where $n = 50p$ samples are available. The entry NA indicates that the corresponding algorithm failed to learn a graph after performing $150,000$ CI tests on average. Each number in the table is obtained using 20 DAGs. In this table, ASC stands for the Average Size of Conditioning sets. Moreover, precision and recall of the learned skeletons along with F1-scores are reported.

The accuracy of all the algorithms is close to each other since the algorithms have access to a large dataset, whereas MARVEL is faster and requires significantly fewer CI tests with smaller conditioning sets compared to the other algorithms.

5.2.3 Scenario 3: real-world structures

Further experimental results are provided with new sets of parameters on real-world structures. Two new structures, namely Alarm and Diabetes are added to the set of structures on which the structure learning algorithms are evaluated.

Table 3 shows the experiment results of this scenario. Each entry of the table is reported as an average of 20 runs, and $n = 15p$ samples are available per variable. The data

---

6. Numbers are in seconds.

(a) Insurance ($|V| = 27, |E| = 51$)

(b) Mildew ($|V| = 35, |E| = 46$)

(c) Barley ($|V| = 48, |E| = 84$)

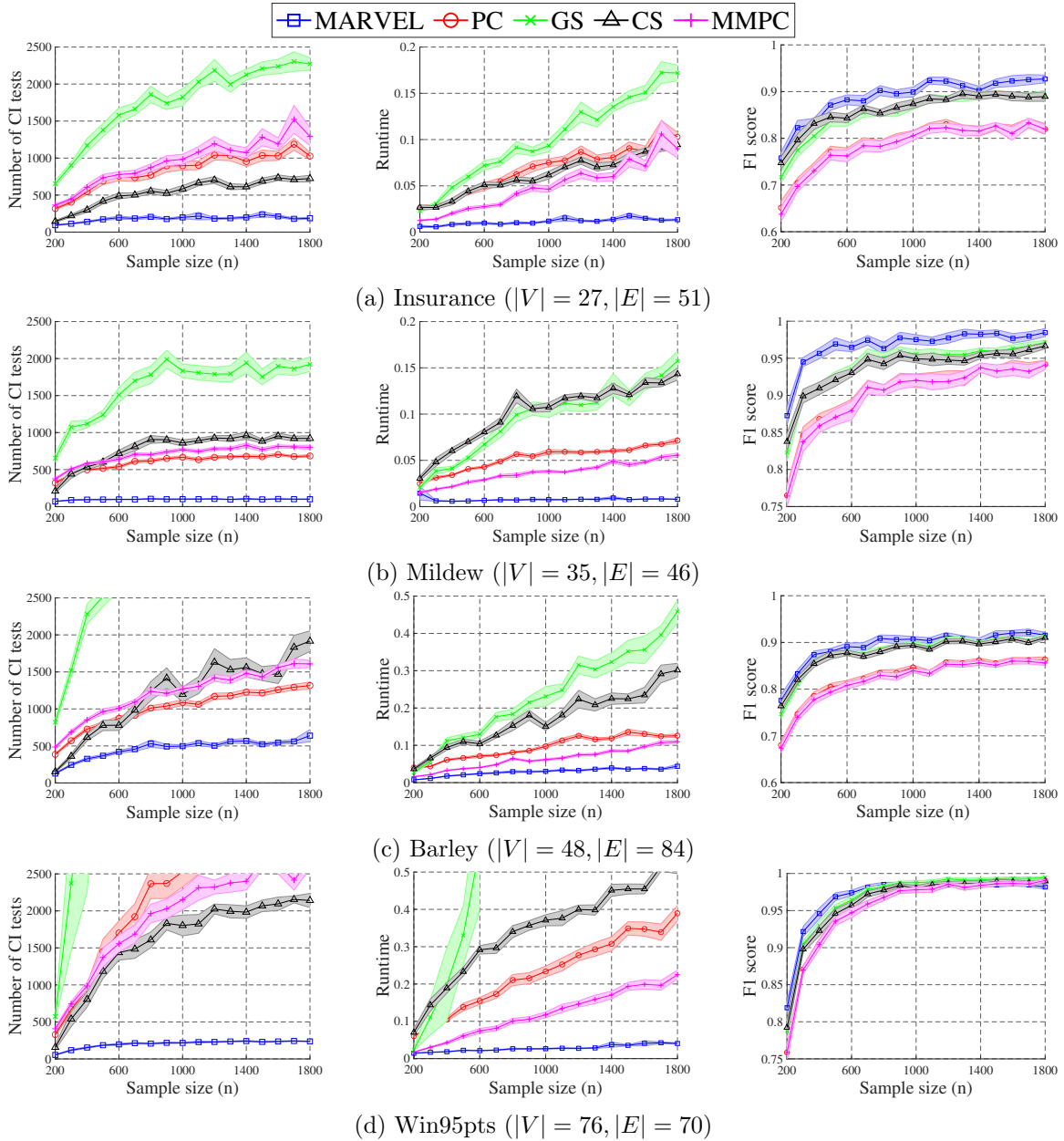(d) Win95pts ($|V| = 76, |E| = 70$)

Figure 4: Scenario 1: The effect of sample size on the performance of the structure learning algorithms for real-world structures after Markov boundary discovery.

generation process is similar to the previous sections, except for the choice of coefficients and the variance of the noise variables. The coefficients are chosen uniformly at random from the interval $[-2, -0.5] \cup [0.5, 2]$ and the noise variables are distributed according to $\mathcal{N}(0, \sigma_X^2)$ where σ_X is chosen uniformly at random from the interval $[1, \sqrt{2}]$.

The experimental results demonstrate that MARVEL significantly outperforms the other algorithms in terms of runtime and the number of the required CI tests while maintaining

Table 2: Scenario 2: Performance of the causal structure learning algorithms on random graphs after Markov boundary discovery ($n = 50p$, $\Delta_{\text{in}} = 4$).

	p	50	60	70	80	90	100	150	200
MARVEL	CI tests	**1,567**	**1,586**	**1,494**	**1,890**	**1,731**	**2,543**	**3,120**	**4,087**
	Runtime	**0.18**	**0.22**	**0.19**	**0.27**	**0.28**	**0.47**	**1.43**	**3.93**
	ASC	2.00	**1.73**	**1.72**	**1.76**	**1.67**	**1.91**	**1.72**	**1.62**
	F1 score	0.93	0.94	0.95	0.95	**0.96**	**0.97**	0.98	0.98
	Precision	0.90	0.92	0.93	0.93	0.94	0.95	0.96	0.96
	Recall	**0.96**	**0.97**	**0.97**	**0.98**	**0.98**	**0.98**	**0.99**	**0.99**
PC	CI tests	2,577	3,113	4,586	5,247	6,094	7,655	12,868	17,643
	Runtime	0.32	0.43	0.71	0.86	1.16	1.51	2.94	5.16
	ASC	**1.80**	1.88	2.16	2.07	2.15	2.29	2.52	2.53
	F1 score	0.92	0.93	0.95	0.95	**0.96**	0.96	**0.98**	**0.98**
	Precision	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	Recall	0.86	0.87	0.90	0.91	0.92	0.93	0.96	0.97
GS	CI tests	61,887	102,296	NA	NA	NA	NA	NA	NA
	Runtime	9.26	16.12	NA	NA	NA	NA	NA	NA
	ASC	5.79	6.19	NA	NA	NA	NA	NA	NA
	F1 score	**0.94**	**0.95**	NA	NA	NA	NA	NA	NA
	Precision	**1.00**	**1.00**	NA	NA	NA	NA	NA	NA
	Recall	0.89	0.90	NA	NA	NA	NA	NA	NA
CS	CI tests	14,091	26,254	27,131	51,522	NA	NA	NA	NA
	Runtime	2.47	4.88	5.45	11.85	NA	NA	NA	NA
	ASC	4.61	5.03	4.94	5.53	NA	NA	NA	NA
	F1 score	**0.94**	0.94	**0.96**	**0.96**	NA	NA	NA	NA
	Precision	0.99	0.99	**1.00**	**1.00**	NA	NA	NA	NA
	Recall	0.89	0.90	0.92	0.93	NA	NA	NA	NA
MMPC	CI tests	2,818	3,467	4,675	5,675	6,412	7,693	12,747	19,250
	Runtime	0.28	0.36	0.51	0.73	0.94	1.17	2.50	4.96
	ASC	2.08	2.13	2.32	2.33	2.37	2.46	2.64	2.78
	F1 score	0.92	0.93	0.94	0.95	**0.96**	0.96	**0.98**	**0.98**
	Precision	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	Recall	0.85	0.87	0.90	0.90	0.92	0.93	0.96	0.97

superior accuracy of the learned graph in most of the experiments. It is worthy to note that one of the caveats of constraint-based methods is the high number of missing edges in the learned structure, whereas MARVEL obtains the highest recall score (i.e., the fewest number of missing edges) as seen in our experimental results. This is due to the fact that MARVEL performs fewer CI tests to verify the existence of an edge. That is, the existence of the edge is assumed if at most $2^{\Delta_{in}-1}$ CI tests yield dependence, whereas this number can grow up to 2^{Δ} or 2^{α} for the other algorithms, which makes them too conservative.

6. Conclusion

We proposed MARVEL, a recursive Markov boundary-based causal structure learning method for efficiently learning the essential graph corresponding to the Markov equivalence class of the causal DAG. We first introduced the notion of removable variables and then designed an efficient algorithm to identify them using Markov boundary information. Then we made use of these variables to learn the causal structure recursively. We showed that MARVEL requires substantially fewer CI tests than the state-of-the-art methods, making it scalable and suitable to be used on systems with a large number of variables. We provided the

Table 3: Scenario 3: Performance of the causal structure learning algorithms on real-world graphs after Markov boundary discovery ($n = 15p$).

Algorithm		Insurance p=27 e=51	Mildew p=35 e=46	Alarm p=37 e=46	Barley p=48 e=84	Win95pts p=76 e=70	Diabetes p=104 e=148
MARVEL	CI tests	**138**	**95**	**95**	**406**	**233**	**296**
	Runtime	**0.01**	**0.01**	**0.01**	**0.03**	**0.03**	**0.07**
	ASC	**0.91**	**0.51**	**0.78**	1.14	**1.19**	**0.57**
	F1 Score	**0.78**	**0.86**	**0.89**	**0.81**	**0.97**	**0.90**
	Precision	0.81	0.89	0.91	0.81	0.96	0.90
	Recall	**0.75**	**0.84**	**0.86**	**0.80**	**0.97**	**0.90**
PC	CI tests	451	403	316	870	1,590	1,584
	Runtime	0.03	0.03	0.03	0.06	0.16	0.25
	ASC	1.02	0.90	0.95	**1.08**	2.21	1.25
	F1 Score	0.62	0.69	0.81	0.73	0.91	0.84
	Precision	0.89	**0.96**	**1.00**	0.96	**1.00**	0.96
	Recall	0.48	0.53	0.68	0.59	0.84	0.74
GS	CI tests	931	1,417	632	3,046	32,821	1,825
	Runtime	0.04	0.05	0.03	0.15	3.29	0.13
	ASC	2.15	2.69	2.03	3.02	6.38	1.62
	F1 Score	0.70	0.76	0.86	0.79	0.94	0.88
	Precision	**0.91**	**0.96**	0.99	**0.98**	**1.00**	**0.98**
	Recall	0.57	0.63	0.76	0.66	0.88	0.80
CS	CI tests	219	665	140	734	1418	445
	Runtime	0.03	0.06	0.03	0.10	0.30	0.27
	ASC	1.56	2.26	1.45	2.15	3.39	1.11
	F1 Score	0.71	0.75	0.86	0.79	0.93	0.87
	Precision	0.90	0.93	0.97	0.95	0.98	0.95
	Recall	0.60	0.63	0.77	0.67	0.89	0.80
MMPC	CI tests	574	563	416	1,114	1,481	2,517
	Runtime	0.02	0.02	0.02	0.04	0.09	0.17
	ASC	1.39	1.23	1.09	1.48	2.14	1.82
	F1 Score	0.61	0.65	0.80	0.71	0.91	0.81
	Precision	0.88	0.92	0.99	0.96	**1.00**	0.88
	Recall	0.47	0.50	0.67	0.57	0.84	0.75

correctness and complexity analyses of the proposed method. We also compared MARVEL with other constraint-based causal structure learning algorithms through various experiments. The results demonstrated the superiority of MARVEL both in terms of complexity and accuracy compared to the other algorithms.

Appendix A. Proofs

We need the following definition in the proofs.

Definition 17 (Descendant) *For vertices X, Y in DAG $\mathcal{G}$, Y is called a* descendant *of X if there is a directed path from X to Y. The set of all descendants of X in $\mathcal{G}$ is denoted by $De(X, \mathcal{G})$. Note that $X \in De(X, \mathcal{G})$.*

A.1 Proofs of Section 3

Remark 4 Suppose $P_{\mathbf{V}}$ is Markov and faithful with respect to a DAG $\mathcal{G}$. For any vertex $X \in \mathbf{V}$, $P_{\mathbf{V} \setminus \{X\}}$ is Markov and faithful with respect to $\mathcal{G} \setminus \{X\}$ if and only if X is a removable vertex in $\mathcal{G}$.
Proof Suppose $Y, Z \in \mathbf{V} \setminus \{X\}$ and $\mathbf{S} \subseteq \mathbf{V} \setminus \{X, Y, Z\}$. By definition, $P_{\mathbf{V} \setminus \{X\}}$ is Markov and faithful with respect to $\mathcal{G} \setminus \{X\}$ if and only if

$$Y \perp_{\mathcal{G} \setminus \{X\}} Z | \mathbf{S} \iff Y \perp\!\!\!\perp_{P_{\mathbf{V} \setminus \{X\}}} Z | \mathbf{S}. \tag{12}$$

By the definition of removability, X is removable in $\mathcal{G}$ if and only if

$$Y \perp_{\mathcal{G} \setminus \{X\}} Z | \mathbf{S} \iff Y \perp_{\mathcal{G}} Z | \mathbf{S}. \tag{13}$$

Since $P_{\mathbf{V}}$ is Markov and faithful with respect to $\mathcal{G}$, the right hand sides of the above equations are equivalent. Hence, the two equations are equivalent. ∎

Theorem 5 (Removability) X is removable in $\mathcal{G}$ if and only if the following two conditions are satisfied for every $Z \in \mathrm{Ch}_X$.

 Condition 1: $N_X \subset N_Z \cup \{Z\}$.

 Condition 2: $\mathrm{Pa}_Y \subset \mathrm{Pa}_Z$ for any $Y \in \mathrm{Ch}_X \cap \mathrm{Pa}_Z$.

Proof

1. To prove the if side, we assume that X is a variable in $\mathcal{G}$ that satisfies Conditions 1 and 2. Let $\mathcal{H} = \mathcal{G} \setminus \{X\}$, the graph obtained by removing X from $\mathcal{G}$. We first prove the following two lemmas.

   Lemma 18 *For any vertex Y of $\mathcal{H}$,*

   $$De(Y, \mathcal{G}) = De(Y, \mathcal{H}).$$

   Proof Suppose $Z \in De(Y, \mathcal{H})$, i.e., there exists a directed path from Y to Z in $\mathcal{H}$. The same path exists in $\mathcal{G}$. Hence, $Z \in De(Y, \mathcal{G})$. Therefore, $De(Y, \mathcal{H}) \subseteq De(Y, \mathcal{G})$.

   Now suppose $Z \in De(Y, \mathcal{G})$, and let $\mathbf{P}$ be a directed path from Y to Z in $\mathcal{G}$. If $\mathbf{P}$ does not include X, it also exists in $\mathcal{H}$. Otherwise, let $\mathbf{P} = (Y = P_1, P_2, ..., P_{i-1}, P_i = X, P_{i+1}, ..., P_k = Z)$. Condition 1 implies that $P_{i-1} \to P_{i+1}$. Hence, $\mathbf{P}' = (Y = P_1, P_2, ..., P_{i-1}, P_{i+1}, ..., P_k = Z)$ is a directed path from Y to Z in $\mathcal{H}$, and $Z \in De(Y, \mathcal{H})$. Therefore, $De(Y, \mathcal{G}) \subseteq De(Y, \mathcal{H})$, which concludes the proof. ∎

Lemma 19 *Let $\mathbf{P}$ be a path in $\mathcal{H}$, and $\mathbf{S}$ be a subset of vertices of $\mathcal{H}$. $\mathbf{S}$ blocks $\mathbf{P}$ in $\mathcal{G}$ if and only if $\mathbf{S}$ blocks $\mathbf{P}$ in $\mathcal{H}$. Moreover, if a vertex blocks $\mathbf{P}$ in one of the graphs, it also blocks $\mathbf{P}$ in the other one.*

Proof The proof of necessary and sufficient conditions are the same. Let $\mathcal{G}_1$ be one of $\mathcal{G}, \mathcal{H}$ and $\mathcal{G}_2$ be the other one. Let Y, Z be the end points of $\mathbf{P}$ and $W \in \mathbf{P}$ be a vertex that blocks $\mathbf{P}$ in $\mathcal{G}_1$. Either $W \in \mathbf{S}$ and W is a non-collider in $\mathbf{P}$, or W is a collider in $\mathbf{P}$ and $\mathrm{De}(W, \mathcal{G}_1) \cap (\mathbf{S} \cup \{Y, Z\}) = \varnothing$. Now consider this path in $\mathcal{G}_2$. In the first case, W is a non-collider included in $\mathbf{S}$ and therefore it blocks $\mathbf{P}$ in $\mathcal{G}_2$ too. In the second case, W is a collider and due to Lemma 18, $\mathrm{De}(W, \mathcal{G}_2) \cap (\mathbf{S} \cup \{Y, Z\}) = \mathrm{De}(W, \mathcal{G}_1) \cap (\mathbf{S} \cup \{Y, Z\}) = \varnothing$. Therefore, W blocks $\mathbf{P}$ in $\mathcal{G}_2$. ■

To show the removability of X, we need to verify Equation 3, i.e., show that for any vertices $Y, Z \in \mathbf{V} \setminus \{X\}$ and $\mathbf{S} \subseteq \mathbf{V} \setminus \{X, Y, Z\}$,

$$Y \perp_{\mathcal{G}} Z | \mathbf{S} \iff Y \perp_{\mathcal{H}} Z | \mathbf{S}.$$

Proving the only if side in Equation 3 is straightforward: Suppose Y and Z are d-separated in $\mathcal{G}$ by $\mathbf{S}$. All paths between Y and Z in $\mathcal{H}$, which are also present in $\mathcal{G}$, are blocked in $\mathcal{G}$ by $\mathbf{S}$. Lemma 19 implies that these paths are also blocked in $\mathcal{H}$ by $\mathbf{S}$. Hence, Y and Z are d-separated in $\mathcal{H}$ by $\mathbf{S}$.

For the reverse direction, suppose Y and Z are d-separated in $\mathcal{H}$ by $\mathbf{S}$. Take an arbitrary path $\mathbf{P}$ between Y and Z in $\mathcal{G}$. We will prove that $\mathbf{P}$ is blocked in $\mathcal{G}$ by $\mathbf{S}$. If $X \notin \mathbf{P}$, then $\mathbf{P}$ is a path in $\mathcal{H}$ too. In this case, since $\mathbf{S}$ is blocking $\mathbf{P}$ in $\mathcal{H}$, Lemma 19 implies that $\mathbf{S}$ blocks $\mathbf{P}$ in $\mathcal{G}$. Otherwise, $X \in \mathbf{P}$. Note that Lemma 19 cannot be used in this case as $\mathbf{P}$ is not a path in $\mathcal{H}$. Suppose $\mathbf{P} = (P_1 = Y, P_2, ..., P_\ell, X, P_r, ..., P_m = Z)$. Two possibilities may occur:

(a) $P_\ell, P_r \in \mathrm{Pa}_X$ (Figure 5a): If there exists a vertex other than X that blocks $\mathbf{P}$ in $\mathcal{H}$, it blocks it in $\mathcal{G}$ too. Otherwise, we need to prove that X is blocking $\mathbf{P}$ in $\mathcal{G}$. X is a collider in $\mathbf{P}$ since $P_\ell, P_r \in \mathrm{Pa}_X$. Note that $X \notin \mathbf{S}$ as X is not present in $\mathcal{H}$. It is left to prove that $\mathrm{De}(X, \mathcal{G}) \cap (\mathbf{S} \cup \{Y, Z\}) = \varnothing$. Let T be an arbitrary descendant of X and $W \in \mathrm{Ch}_X$ be the first vertex on a directed path from X to T (it might happen that $T = W$). As X satisfies Condition 1, P_ℓ, P_r are connected to W. Now consider $\mathbf{P}' = (P_1, ..., P_\ell, W, P_r, ..., P_m)$ which is a path between Y and Z in $\mathcal{H}$. $\mathbf{S}$ blocks this path in $\mathcal{H}$, but none of the vertices of $\mathbf{P}'$ except for W can block $\mathbf{P}'$ in $\mathcal{H}$. This is because otherwise according to Lemma 19, the same vertex would block $\mathbf{P}$ in $\mathcal{G}$ too, which is against the assumption. Hence, W blocks $\mathbf{P}'$. W is a collider in $\mathbf{P}'$ and therefore, $\mathrm{De}(W, \mathcal{G}) \cap (\mathbf{S} \cup \{Y, Z\}) = \varnothing$. This proves that $T \notin (\mathbf{S} \cup \{Y, Z\})$. As a result, $\mathbf{P}$ is blocked in $\mathcal{G}$ by $\mathbf{S}$.

(b) $P_\ell \in \mathrm{Ch}_X$ or $P_r \in \mathrm{Ch}_X$ (Figure 5b): We assume without loss of generality that P_r appears later than P_ℓ in the causal order. Therefore, $P_r \in \mathrm{Ch}_X$. Due to Condition 1, $P_\ell \to P_r$ is an edge in $\mathcal{H}$. Hence, $\mathbf{P}' = (P_1, ..., P_\ell, P_r, ..., P_m)$ is a path between Y and Z in $\mathcal{H}$. This path is blocked in $\mathcal{H}$ (and also in $\mathcal{G}$ due to Lemma 19) by a vertex W. If $W \neq P_\ell$, or $W = P_\ell$ and P_ℓ is a non-collider in $\mathbf{P}$, W blocks $\mathbf{P}$ in $\mathcal{G}$. The only remaining case to consider is when P_ℓ blocks $\mathbf{P}'$

19

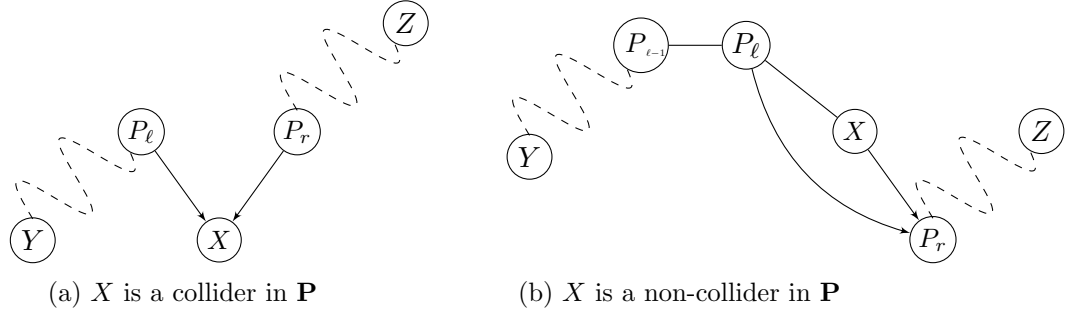(a) X is a collider in $\mathbf{P}$ (b) X is a non-collider in $\mathbf{P}$

Figure 5: Omitting a removable vertex.

but it is a collider in $\mathbf{P}$. In this case, $P_\ell \in \mathbf{S}$ since it is a non-collider in $\mathbf{P}'$. Moreover, $P_{\ell-1}$ is a parent of P_ℓ and due to Condition 2, $P_{\ell-1} \to P_r$ is an edge in $\mathcal{H}$. Hence, $\mathbf{P}'' = (P_1, ..., P_{\ell-1}, P_r, ..., P_m)$ is a path between Y and Z in $\mathcal{H}$. Now the same vertex that blocks $\mathbf{P}''$ in $\mathcal{H}$, blocks $\mathbf{P}$ in $\mathcal{G}$. Note that in this case $P_{\ell-1}$ is not a collider in either $\mathbf{P}''$ or $\mathbf{P}$.

In both cases $\mathbf{P}$ is blocked in $\mathcal{G}$ by $\mathbf{S}$. Hence, equation 3 holds and X is removable.

2. To prove the only if side of Theorem 5, it suffices to show that if X is removable, it then satisfies conditions 1 and 2.

   Condition 1: Suppose $Z \in \mathrm{Ch}_X$ and $W \in N_X$. Either $W \leftarrow X \to Z$ or $W \to X \to Z$ is a path in $\mathcal{G}$. A set $\mathbf{S}$ can block such paths only if $X \in \mathbf{S}$. Since $X \notin \mathcal{H}$, no separating set for W and Z exists in $\mathcal{H}$. Equation 3 implies that no separating set for W and Z exists in $\mathcal{G}$. Therefore, W, Z must be connected with an edge and X satisfies Condition 1.

   Condition 2: Suppose $Z \in \mathrm{Ch}_X$, $Y \in \mathrm{Ch}_X \cap \mathrm{Pa}_Z$, and $U \in \mathrm{Pa}_Y$. We show that no $\mathbf{S}$ can d-separate U and Z in $\mathcal{G}$: $U \to Y \to Z$ and $U \to Y \leftarrow X \to Z$ are both paths in $\mathcal{G}$. $\mathbf{S}$ can block the first path only if $Y \in \mathbf{S}$. But then the latter path can only be blocked if $X \in \mathbf{S}$. Since $X \notin \mathcal{H}$, no such $\mathbf{S}$ exists in $\mathcal{H}$. Hence, Equation 3 implies that no separating set for U and Z exists in $\mathcal{G}$. Therefore, U, Z must be connected with an edge. Since $U \to Y$ and $Y \to Z$, U must be a parent of Z and X satisfies Condition 2.

$\blacksquare$

Lemma 7 Suppose $X \in \mathbf{V}$ and $Y \in Mb_X$. Y is a neighbor of X if and only if

$$X \not\perp\!\!\!\perp Y | \mathbf{S}, \quad \forall \mathbf{S} \subset Mb_X \setminus \{Y\}. \tag{14}$$

Proof If $Y \in N_X$, then X, Y do not have any separating set. Otherwise, suppose $Y \notin N_X$. It suffices to find a $\mathbf{S} \subset Mb_X \setminus \{Y\}$ that d-separates X, Y. By local Markov property, if Y is not a descendant of X, $\mathbf{S} = Pa_X$ would do. Now suppose Y is a descendant of X. Define $\mathbf{S}$ as the set of vertices in Mb_X that appear earlier than Y in the causal order. We claim $X \perp\!\!\!\perp Y | \mathbf{S}$, i.e., $\mathbf{S}$ blocks all the paths between X and Y. Take an arbitrary path $\mathbf{P}$ between X and Y and let $Z \in \mathbf{P}$ be the latest vertex of $\mathbf{P}$ in the causal order. Two cases may occur:

1. $Z = Y$: In this case, all vertices on $\mathbf{P}$ appear earlier than Y in the causal order and $\mathbf{P} \cap Mb_X \subseteq \mathbf{S}$. Now let the two vertices following X on $\mathbf{P}$ be W_1, W_2 ($\mathbf{P}$ has a length of at least 2 as Y is not a neighbor of X). If W_1 is a parent of X, it is included in $\mathbf{S}$ and it blocks $\mathbf{P}$. Otherwise, W_1 is a child of X. Now either $W_2 \in \mathrm{Ch}_{W_1}$ and W_1 blocks $\mathbf{P}$ or $W_2 \in \mathrm{Pa}_{W_1}$ and W_2 blocks $\mathbf{P}$. Note that in the latter case W_2 is a parent of a child of X and is included in Mb_X. Therefore, $W_2 \in \mathbf{S}$.

2. $Z \neq Y$: In this case Z is a collider on $\mathbf{P}$ because both vertices before and after Z appear earlier in the causal order and are therefore parents of Z. Due to the definition of $\mathbf{S}$, neither Z nor any of its descendants are in $\mathbf{S} \cup \{X, Y\}$. Hence, $\mathbf{P}$ is blocked by $\mathbf{S}$.

In all of the cases, the introduced $\mathbf{S}$ is not equal to $Mb_X \setminus \{Y\}$ as it does not include the common child of X and Y. ∎

Lemma 8 Suppose $T \in \Lambda_X$ with a separating set $\mathbf{S}_{XT}$ for X and T, and let $Y \in N_X$. Y is a common child of X and T (i.e., $X \to Y \leftarrow T$ is in $\mathcal{V}_X^{\mathrm{pa}}$) if and only if $Y \notin \mathbf{S}_{XT}$ and

$$Y \not\perp\!\!\!\perp T | \mathbf{S}, \quad \forall \mathbf{S} \subseteq Mb_X \cup \{X\} \setminus \{Y, T\}. \tag{15}$$

Proof Suppose $Y \in N_X$ is a common child of X and T. $\mathbf{S}_{XT}$ blocks the path $X \to Y \leftarrow T$ between X, T. Hence, Y cannot be in $\mathbf{S}_{XT}$. Additionally, Y, T are neighbors and they do not have any separating set.

Now suppose $Y \in N_X$ is not a common child of X and T, and $Y \notin \mathbf{S}_{XT}$. It suffices to find a $\mathbf{S} \subseteq Mb_X \cup \{X\} \setminus \{Y, T\}$ that d-separates Y and T. First, note that Y and T cannot be neighbors because otherwise, (X, Y, T) is a path between X, T that must be blocked by $\mathbf{S}_{XT}$, but Y is not a collider in this path, and $Y \notin \mathbf{S}_{XT}$ which is not possible. In order to introduce $\mathbf{S}$, consider three possible cases for Y:

- $Y \in \mathrm{Pa}_X$: We claim $\mathbf{S} = \mathbf{S}_{XT}$ d-separates Y and T. Let $\mathbf{P}$ be a path between Y and T. If $\mathbf{P}$ includes X, it is already blocked by $\mathbf{S}_{XT}$ as there exists a vertex that blocks the part of the path between X and T, and the same vertex blocks $\mathbf{P}$. If $\mathbf{P}$ does not include X, let $\mathbf{P}' = (X, Y, \ldots, T)$, i.e., the path from X to T through Y and $\mathbf{P}$. $\mathbf{P}'$ is blocked by a vertex Z. As $Y \notin \mathbf{S}$ and Y is not a collider in $\mathbf{P}'$, $Z \neq Y$. Therefore, Z blocks $\mathbf{P}$ too.

- $Y \in \mathrm{Ch}_X$ and T is a descendant of Y: Similarly, $\mathbf{S} = \mathbf{S}_{XT}$ d-separates Y and T. To prove this claim, let $\mathbf{P}$ be a path between Y and T. If $\mathbf{P}$ includes X, with the same statements discussed above, $\mathbf{P}$ is blocked by $\mathbf{S}$. If $\mathbf{P}$ does not include X, define $\mathbf{P}' = (X, Y, \ldots, T)$, and let Z be the vertex that blocks $\mathbf{P}'$. $Z \neq Y$ because $Y \notin \mathbf{S}$ and Y cannot block $\mathbf{P}'$ as a collider since T is a descendant of Y. Hence, Z blocks $\mathbf{P}$ too.

- $Y \in \mathrm{Ch}_X$ and T is not a descendant of Y: In this case, the set of parents of Y d-separate Y and T. Note that as Y is a child of X, $\mathrm{Pa}_Y \in Mb_X \cup \{X\}$ and $\mathbf{S} = \mathrm{Pa}_Y$ is our desired set.

In all the cases we introduced a $\mathbf{S} \subseteq Mb_X \cup \{X\} \setminus \{Y, T\}$ that d-separates Y and T. ∎

Lemma 9 Variable X satisfies Condition 1 of Theorem 5 if and only if

$$Z \not\perp\!\!\!\perp W | \mathbf{S} \cup \{X\}, \quad \forall W, Z \in N_X, \mathbf{S} \subseteq Mb_X \setminus \{Z, W\}. \tag{16}$$

Proof Suppose X satisfies condition 1, and let Z, W be two of its neighbors. If at least one of Z, W is a child of X, then condition 1 implies that Z and W are neighbors and cannot be d-separated with any set. If both of Z, W are parents of X, no set including X can d-separates Z, W since X is a collider in $Z \to X \leftarrow W$.

For the if side, suppose that $Z \not\perp\!\!\!\perp W | \mathbf{S} \cup \{X\}$ for any pair of vertices $W, Z \in N_X$ and any $\mathbf{S} \subseteq Mb_X \setminus \{Z, W\}$. We need to show that X satisfies condition 1. Let Z, W be two arbitrary vertices in N_X where at least one of them is a child of X. It suffices to show that W and Z are neighbors. Without loss of generality we can assume that Z appears later than W in the causal order. Therefore $Z \in \mathrm{Ch}_X$. Assume by contradiction that they are not neighbors. Since W is not a descendant of Z, local Markov property implies that the set of parents of Z d-separates W and Z. Note that Z is a child of X and therefore $\mathrm{pa}_Z \subseteq Mb_X \cup \{X\}$. Hence, $\mathbf{S} = \mathrm{Pa}_Z \setminus \{X\} \subseteq Mb_X \setminus \{Z, W\}$ would be a set that $Z \perp\!\!\!\perp W | \mathbf{S} \cup \{X\}$ which is against our assumption. Therefore, W and Z are neighbors and X satisfies condition 1. ∎

Lemma 10 Suppose the variable X satisfies Condition 1 of Theorem 5. Then X satisfies Condition 2 of Theorem 5, and therefore, X is removable, if and only if

$$Z \not\perp\!\!\!\perp T | \mathbf{S} \cup \{X, Y\}, \quad \forall (X \to Y \leftarrow T) \in \mathcal{V}_X^{\mathrm{pa}}, Z \in N_X \setminus \{Y\}, \mathbf{S} \subseteq Mb_X \setminus \{Z, Y, T\}. \tag{17}$$

Proof Suppose X is removable. Let $X \to Y \leftarrow T$ be a v-structure and $Z \in N_X$. If Z appears later than Y in the causal order, then condition 1 implies that $Y \in \mathrm{Pa}_Z$. Therefore, condition 2 implies that $T \in \mathrm{Pa}_Z$. Hence, Z and T are neighbors and cannot be d-separated. If Z appears earlier than Y in the causal order, condition 1 implies that $Z \in \mathrm{Pa}_Y$. In this case $Z \to Y \leftarrow T$ is a path between Z, T and no set containing Y can d-separate Z, T.

For the if side, suppose X satisfies condition 1 and the assumption of the lemma holds. We need to show that X satisfies condition 2. Suppose $Z \in \mathrm{Ch}_X$, $Y \in \mathrm{Ch}_X \cap \mathrm{Pa}_Z$, and $T \in \mathrm{Pa}_Y$. It suffices to show that T, Z are neighbors. Assume by contradiction that Z, T are not neighbors. Note that $X, Y \in \mathrm{Pa}_Z$. Define $\mathbf{S} = \mathrm{Pa}_Z \setminus \{X, Y\}$. Parents of Z are in Mb_X, and Z, T are not neighbors. Therefore, $\mathbf{S} \subseteq Mb_X \setminus \{Z, Y, T\}$. Since T is not a descendant of Z (in fact, Z is a descendant of T), parents of Z d-separates Z, T. Hence, $\mathbf{S} \cup \{X, Y\}$ d-separates Z, T which is against the assumption of the lemma. Hence, Z, T are neighbors and X is removable. ∎

Lemma 12 If $X \in \mathbf{V}$ is a removable vertex in $\mathcal{G}$, then $|Mb_X| \leq \Delta_{\mathrm{in}}$, where Δ_{in} is the maximum in-degree of $\mathcal{G}$.

Proof Let Z be the latest child of X and therefore, the latest vertex of Mb_X in the causal order. From Theorem 5, every vertex in $Mb_X \setminus \{Z\}$ is connected to Z. By definition of Z, vertices in $Mb_X \setminus \{Z\}$ along with X itself must be the parents of Z, that is $\{X\} \cup Mb_X \setminus \{Z\} = \mathrm{Pa}_Z$. The cardinality of the left hand side is equal to $|Mb_X|$, while the cardinality of the right hand side is bounded by Δ_{in}. Hence,

$$|Mb_X| = |\mathrm{Pa}_Z| \leq \Delta_{\mathrm{in}}.$$

∎

22

Theorem 13 (Correctness of MARVEL) Suppose $\mathcal{G}$ satisfies Markov property and faithfulness with respect to $P_{\mathbf{V}}$. The learned graph $\hat{\mathcal{G}}$ in Algorithm 1 has the same skeleton and v-structures as $\mathcal{G}$. Therefore, the output of Algorithm 1 is the essential graph corresponding to $\mathcal{G}$.

Proof First, note that as it is discussed in Section 3.3, Markov and faithfulness assumptions hold in all iterations of Algorithm 1. Hence, throughout the execution of the algorithm, the result of CI tests are equivalent to the d-separation relations in the remaining graph. Further note that Remark 6 implies that there always exists at least one removable variable at each iteration. Therefore, Algorithm 1 never gets stuck. We now show that $\hat{\mathcal{G}}$ has the same skeleton and the same set of v-structures as $\mathcal{G}$.

- *Skeleton*: We show $\hat{\mathcal{G}}$ contains all the edges of $\mathcal{G}$ and it has no extra edges.

  No false positives: The algorithm starts with an empty graph. At each iteration, new edges are only added between $X_{(i)}$ and its neighbors (line 8). Hence, no extra edges appear in $\hat{\mathcal{G}}$.

  No false negatives: Suppose there exists an edge between X and Y in $\mathcal{G}$. Without loss of generality, suppose X gets removed before Y. During the iteration at which the algorithm removes X, it identifies all the remaining neighbors of X, including Y, and adds an edge between X and its neighbors. Therefore, the skeleton of $\hat{\mathcal{G}}$ contains the edge between X and Y.

- *V-structures*: It suffices to show that for every X, Y and Z,

  $X \to Y \leftarrow Z$ is a v-structure in $\mathcal{G}$ if and only if $X \to Y \leftarrow Z$ is a v-structure in $\hat{\mathcal{G}}$.

  If side: Suppose $X \to Y \leftarrow Z$ is a v-structure in $\hat{\mathcal{G}}$. Consider the two corresponding edges from X to Y and Z to Y. Each edge is either oriented in line 11 of the algorithm, where it is a part of a v-structure. In this case, it is oriented correctly due to Lemma 8. Otherwise, it is oriented in line 13 when vertex Y gets removed. Since Y is getting removed, it is removable while both X and Z are present in the remaining graph. Therefore, Y satisfies condition 1 of Theorem 5. We know from the first part of the proof that $\hat{\mathcal{G}}$ has the same skeleton as $\mathcal{G}$. Hence, X and Z are not neighbors in $\mathcal{G}$. Neither X nor Z can be a child of Y in $\mathcal{G}$, since otherwise, condition 1 implies that X and Z are neighbors. Therefore, they must both be parents of Y in $\mathcal{G}$. That is, $X \to Y \leftarrow Z$ is a v-structure in $\mathcal{G}$.

  Only if side: Suppose $X \to Y \leftarrow Z$ is a v-structure in $\mathcal{G}$. We show that the edge between X and Y is oriented from X to Y in $\hat{\mathcal{G}}$. We orient every edge either in line 11 where this edge appears in an identified v-structure or in line 13 when the edge is undirected and we want to remove one of its endpoints. Note that all the edges are oriented during the algorithm. In the first case, the orientation of the edge is correct due to Lemma 8. In the latter case, i.e., if the edge between X and Y is oriented in line 13, one of the endpoints is discovered to be removable. It suffices to show that the removable endpoint is Y, and hence, the edge is oriented from X to Y. Suppose the opposite, that is X is discovered to be removable and the algorithm reaches line 13. In this case, the v-structure $X \to Y \leftarrow Z$ must be identified in line

10 as a member of $\mathcal{V}_X^{\mathrm{pa}}$, and the edge between X and Y is oriented in line 11, which contradicts the assumption that this edge is not oriented until line 13. Therefore, the removable endpoint is Y, and while Y is removed, the undirected edge is oriented correctly from X to Y.

In both cases, the edge is from X to Y. The same arguments hold for the edge between Z and Y. Therefore, $X \to Y \leftarrow Z$ is a v-structure in $\hat{\mathcal{G}}$.

■

A.2 Proofs of Section 4

Proposition 14 Given the initial Markov boundaries, the number of CI tests required by Algorithm 1 on a graph of order p and maximum in-degree Δ_{in} is upper bounded by

$$p\binom{\Delta_{\mathrm{in}}}{2} + \frac{p}{2}\Delta_{\mathrm{in}}(1 + 0.45\Delta_{\mathrm{in}})2^{\Delta_{\mathrm{in}}} = \mathcal{O}(p\Delta_{in}^2 2^{\Delta_{in}}). \tag{18}$$

Proof MARVEL performs CI tests for the following purposes:

1. *Updating Markov boundaries at the end of each iteration*: As discussed in Section 3.3.1, when X is removed, it is enough to perform a CI test for each pair $(Y, Z) \in N_X \times N_X$. There are $\binom{|N_X|}{2}$ such pairs and from Lemma 12 we know $|N_X| \leq |Mb_X| \leq \Delta_{\mathrm{in}}$. Hence, at most $p\binom{\Delta_{\mathrm{in}}}{2}$ CI tests are performed for updating the Markov boundaries throughout the algorithm.

2. *Testing for removability*: As discussed in Section 3.2, given Markov boundary information, for each variable X we can test its removability by first finding N_X, Λ_X, then finding $\mathcal{V}_X^{\mathrm{pa}}$, and then checking Conditions 1 and 2 of Theorem 5. We showed that we can do this by performing at most

$$K = |Mb_X|2^{|Mb_X|-1} + |\Lambda_X||N_X|2^{|Mb_X|-1} + \binom{|N_X|}{2}2^{|Mb_X|-2} + |\Lambda_X||N_X|2^{|Mb_X|-2}$$

$$= 2^{|Mb_X|-2}(2|Mb_X| + 3|\Lambda_X||N_X| + \binom{|N_X|}{2})$$

CI tests. From Lemma 12 we know $|Mb_X| \leq \Delta_{\mathrm{in}}$. Suppose $N = |N_X|$. Since $|Mb_X| = |N_X| + |\Lambda_X|$ we have:

$$K < 2^{2\Delta_{\mathrm{in}}-2}(2\Delta_{\mathrm{in}}+3(\Delta_{\mathrm{in}}-N)N+\frac{N^2}{2}) \leq 2^{\Delta_{\mathrm{in}}-2}(2\Delta_{\mathrm{in}}+0.9\Delta_{\mathrm{in}}^2) = \frac{1}{2}\Delta_{\mathrm{in}}(1+0.45\Delta_{\mathrm{in}})2^{\Delta_{\mathrm{in}}}$$

As we discussed in Section 3.4, we only need to perform CI tests for testing the removability of a variable for the first time. Because we can save some information and avoid performing new CI tests in the next iterations. Hence, at most $\frac{p}{2}\Delta_{\mathrm{in}}(1 + 0.45\Delta_{\mathrm{in}})2^{\Delta_{\mathrm{in}}}$ CI tests are performed for this task.

Summing up over the above bounds, we get the desired upper bound.

∎

Theorem 16 The number of conditional independence tests of the form $X \perp\!\!\!\perp Y|\mathbf{S}$ required by any constraint-based algorithm on a graph of order p and maximum in-degree Δ_{in} in the worst case is lower bounded by

$$\Omega(p^2 + p\Delta_{in}2^{\Delta_{in}}). \tag{19}$$

Proof We first present an example that requires at least

$$\lfloor\frac{p}{\Delta_{\text{in}}+1}\rfloor\binom{\Delta_{\text{in}}+1}{2}2^{\Delta_{\text{in}}-1} = \Omega(p\Delta_{in}2^{\Delta_{in}}) \tag{20}$$

CI tests to uniquely find its skeleton.

  Example 1. Let $d = \Delta_{\text{in}}$. Suppose the variables are denoted by $X_1, X_2, ..., X_p$ and the first $(d+1)\lfloor\frac{p}{d+1}\rfloor$ variables are partitioned into $\lfloor\frac{p}{d+1}\rfloor$ clusters $C_1, ..., C_{\lfloor\frac{p}{d+1}\rfloor}$ each of size $(d+1)$. Let $\mathcal{G}$, the causal graph, have the following structure: X_i is the i-th vertex in the causal order, the induced sub-graph over the vertices of each cluster is a complete graph, and there is no edge between vertices of different clusters. Note that the maximum in-degree of $\mathcal{G}$ is d. Given any algorithm $\mathcal{A}$ that performs fewer CI tests than the claimed lower bound in Equation 20, we provide a graph $\mathcal{H}$ such that $\mathcal{A}$ fails to tell $\mathcal{G}$ and $\mathcal{H}$ apart.

  Considering the structure of $\mathcal{G}$, for an arbitrary $\mathbf{S}$, any CI test of X_i and X_j conditioned on $\mathbf{S}$ that $\mathcal{A}$ queries, yields dependence if X_i and X_j are in the same cluster and yields independence otherwise. There are $M = \lfloor\frac{p}{d+1}\rfloor\binom{d+1}{2}$ pairs $\{X_i, X_j\}$ such that X_i and X_j are in the same cluster. As algorithm $\mathcal{A}$ performs less than $M2^{d-1}$ CI tests, there exists a pair $\{X_{i^*}, X_{j^*}\}$ in a particular cluster for which algorithm $\mathcal{A}$ queries the conditional independence of $\{X_{i^*}$ and $X_{j^*}\}$ conditioned on fewer than 2^{d-1} sets. Without loss of generality, suppose $i^* < j^* \leq d+1$ and the corresponding cluster is $C_1 = \{X_1, ..., X_{d+1}\}$. As $C_1 \setminus \{X_{i^*}, X_{j^*}\}$ has 2^{d-1} subsets, there exists at least one subset $\mathbf{S}^* \subseteq C_1 \setminus \{X_{i^*}, X_{j^*}\}$ such that for no $\mathbf{S}' \subseteq \mathbf{V} \setminus C_1$, algorithm $\mathcal{A}$ queries the result of the CI test $X_{i^*} \perp\!\!\!\perp X_{j^*}|\mathbf{S}$ where $\mathbf{S} = \mathbf{S}^* \cup \mathbf{S}'$.

  Now we provide the graph $\mathcal{H}$ as follows. It has the same structure and causal order as $\mathcal{G}$, except over the vertices of C_1. As depicted in Figure 6, the vertices in $\mathbf{S}^*$ in an arbitrary order, form the first vertices of C_1 in the causal order. These vertices are followed by X_i^* and X_j^* and then the rest of the vertices of C_1, again in an arbitrary order. As for the skeleton, all pairs in C_1 are connected to each other except for (X_i^*, X_j^*).

  It suffices to show that $\mathcal{H}$ is consistent with all the CI tests that $\mathcal{A}$ performs. For an arbitrary CI test between X_i and X_j conditioned on $\mathbf{S}$, if $\{X_i, X_j\} \neq \{X_{i^*}, X_{j^*}\}$ the test yields the same result for $\mathcal{G}$ and $\mathcal{H}$. If $\{X_i, X_j\} = \{X_{i^*}, X_{j^*}\}$, the result always yields dependence as there is an edge between X_i^* and X_j^* in $\mathcal{G}$. To prove the consistency, we have to show that none of the sets $\mathbf{S}$ among the conditioning sets of the CI tests that $\mathcal{A}$ performs, d-separates X_i^* and X_j^*. The structure of $\mathcal{H}$ implies that $\mathbf{S}^*$ is the unique subset of C_1 which d-separates X_i^* and X_j^*. Since there is no edge between vertices of different clusters, if $\mathbf{S}$ d-separates X_i^* and X_j^* in $\mathcal{H}$ then $\mathbf{S} \cap C_1$ must be equal to $\mathbf{S}^*$. As mentioned above, there is no such CI test performed in $\mathcal{A}$. Therefore, $\mathcal{H}$ is consistent with the results of the CI tests, and $\mathcal{A}$ cannot uniquely determine the skeleton of $\mathcal{G}$.
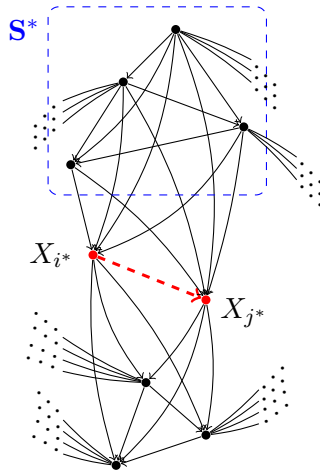
Figure 6: Cluster C_1 in the graph $\mathcal{H}$

We now provide another example that requires $\Omega(p^2)$ CI tests to uniquely find its skeleton.

Example 2. Suppose the causal graph $\mathcal{G}$ is an empty graph with p vertices. Given any algorithm $\mathcal{A}$ that performs fewer CI tests than $\binom{p}{2}$, we provide a graph $\mathcal{H}$ such that $\mathcal{A}$ fails to tell $\mathcal{G}$ and $\mathcal{H}$ apart.

As algorithm $\mathcal{A}$ performs less than $\binom{p}{2}$ CI tests, there exists a pair $\{X_{i^*}, X_{j^*}\}$ for which algorithm $\mathcal{A}$ does not query the conditional independence of X_{i^*} and X_{j^*} conditioned any set. Let $\mathcal{H}$ be the graph with p vertices and only one edge (X_{i^*}, X_{j^*}). Note that, all of the performed CI tests yield independence since $\mathcal{G}$ is an empty graph. Therefore, $\mathcal{H}$ is consistent with the results of the CI tests, and $\mathcal{A}$ cannot uniquely determine the skeleton of $\mathcal{G}$.

Combining the lower bounds of the above examples, we get the desired lower bound of Equation 19.

∎

References

Rob Campbell. Shaded error bar for matlab, 2020. https://github.com/raacampbell/shadedErrorBar.

David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

G Elidan. Bayesian Network Repository, 2001. https://www.cse.huji.ac.il/ galel/Repository/.

Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

Ronald A Fisher. Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population. *Biometrika*, 10(4):507–521, 1915.

Shunkai Fu and Michel C Desmarais. Markov blanket based feature selection: A review of past decade. In *Proceedings of the world congress on engineering*, volume 1, pages 321–328. Newswood Ltd, 2010.

Clark N Glymour and Gregory F Cooper. *Computation, causation, and discovery*. AAAI Press, 1999.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.

Patrik O Hoyer, Dominik Janzing, Joris M Mooij, Jonas Peters, and Bernhard Schölkopf. Nonlinear causal discovery with additive noise models. In *Advances in neural information processing systems*, pages 689–696, 2009.

Markus Kalisch and Peter Bühlmann. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8(Mar):613–636, 2007.

Hasan Manzour, Simge Küçükyavuz, Hao-Hsiang Wu, and Ali Shojaie. Integer programming for learning directed acyclic graphs from continuous data. *Informs Journal on Optimization*, 3(1):46–73, 2021.

Dimitris Margaritis and Sebastian Thrun. Bayesian network induction via local neighborhoods. *Advances in Neural Information Processing Systems*, 12:505–511, 1999.

Christopher Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 403–410, 1995.

Kevin Murphy et al. The bayes net toolbox for matlab. *Computing science and statistics*, 33(2):1024–1034, 2001.

Kevin P Murphy and Ido Rosen. *The bayes net toolbox for matlab*, 2019. URL `https://github.com/bayesnet/bnt/tree/master/BNT/learning`.

Richard E Neapolitan. *Learning Bayesian networks*, volume 38. Pearson Prentice Hall Upper Saddle River, NJ, 2004.

Judea Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference.* Morgan Kaufmann, 1988.

Judea Pearl. *Causality.* Cambridge university press, 2009.

Jean-Philippe Pellet and André Elisseeff. Using markov blankets for causal structure learning. *Journal of Machine Learning Research*, 9(Jul):1295–1342, 2008.

Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, and Antti Kerminen. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7 (Oct):2003–2030, 2006.

Liam Solus, Yuhao Wang, and Caroline Uhler. Consistency guarantees for greedy permutation-based causal inference algorithms. *arXiv preprint arXiv:1702.03530*, 2017.

Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search.* MIT press, 2000.

Marc Teyssier and Daphne Koller. Ordering-based search: A simple and effective algorithm for learning bayesian networks. *arXiv preprint arXiv:1207.1429*, 2012.

Ioannis Tsamardinos and Constantin F Aliferis. Towards principled feature selection: Relevancy, filters and wrappers. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

Ioannis Tsamardinos, Constantin F Aliferis, and Alexander Statnikov. Time and sample efficient discovery of markov blankets and direct causal relations. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 673–678, 2003a.

Ioannis Tsamardinos, Constantin F Aliferis, Alexander R Statnikov, and Er Statnikov. Algorithms for large scale markov blanket discovery. In *FLAIRS conference*, volume 2, pages 376–380, 2003b.

Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

Konstantinos Tsirlis, Vincenzo Lagani, Sofia Triantafillou, and Ioannis Tsamardinos. *M3HC Matlab Implementation*, 2018. URL `https://github.com/mensxmachina/M3HC`.

Thomas Verma and Judea Pearl. *Equivalence and synthesis of causal models.* UCLA, Computer Science Department, 1991.

Xianchao Xie and Zhi Geng. A recursive method for structural learning of directed acyclic graphs. *Journal of Machine Learning Research*, 9(Mar):459–483, 2008.

Sandeep Yaramakala and Dimitris Margaritis. Speculative markov blanket discovery for optimal feature selection. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4–pp. IEEE, 2005.

Hao Zhang, Shuigeng Zhou, Chuanxu Yan, Jihong Guan, and Xin Wang. Recursively learning causal structures using regression-based conditional independence test. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3108–3115, 2019.

Kun Zhang, Bernhard Schölkopf, Peter Spirtes, and Clark Glymour. Learning causality and causality-related learning: Some recent progress. *National Science Review*, 5(1):26–29, 2017.

Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. *arXiv preprint arXiv:1803.01422*, 2018.